Remarks

This is filed in response to the final Office Action, mailed April 19, 2004, rejecting the pending claims over 35 USC 101 (claims 36 – 38), 35 USC 112 (claims 36 – 38), 35 USC 102 (claims 36 –38 and 53) and 35 USC 103 (claims 27 – 30, 34, 39 – 41, 43 –44). The principal reference is Lipkin (US 2002/0049788A1). The secondary references for the rejection under 35 USC 103 are Bradford (US 6,678,679) and Delcambre (US 2002/0059566).

The claims are amended for matters of form, above. Entry of those amendments is therefore requested. The claimed subject matter is shown as being patentably distinct from the cited art, below. Withdrawal of the rejections is therefore requested.

Formal Amendments

At the outset, claims 34 and 41 are amended to remove multiple dependencies. This is made without prejudice, the Applicant reserving the right to present the cancelled subject matter at a later time, e.g., in a continuation hereof. Claims 36 –38 are likewise cancelled without prejudice.

Claim 53 is Not Anticipated by Lipkin

Claim 53 recites a digital data processing method for enterprise application integration including removing redundancies in RDF triples by executing the steps of (i) comparing sequential levels of objects of RDF triples, (ii) determining a confidence level that two or more triplets at a compared level represent redundant information, and (iii) merging into a bag triplets determined to be redundant on a basis of that confidence level.

Lipkin fails to teach these steps. Specifically, with respect to that part of claim 53 which recites "determining a confidence level that two or more triplets at a compared level represent redundant information," the Examiner cites Lipkin at p. 71, col. 1, lines 27 – col. 2, line 25. That text is reprinted below. Nowhere does it suggest removing redundancy — much less, doing so based on confidence levels. Indeed, quite the contrary, in the cited passage, Lipkin laments that his chosen schema is redundant, yet, he proposes no solution:

[1103] Database schema

[1104] The database schema used has two main advantages:

[1105] 1. Simplicity. All RDF data is stored in a single table and all SQL is written to read and write to this table.

[1106] 2. Support for non-RDF data. It is simple to cast non-RDF data into this format so that existing or legacy data can be queried by the DatabaseMR using RQL.

[1107] So, for example, for the following example data stored in an "invoices" table:

id	last_updated	customer
1	10-JAN-99	Ford
2	25-FEB-99	Cisco

[1108] The view used by the MR can be augmented as followed to incorporate this data:

```
create view invoice_date_triples as
select 'last_updated' "rdf_property",
       ('invoice#' || id) "rdf_resource",
       to_char(last_updated, 'YYYY-MM-DD') "rdf_object"
       from test_invoices;
create view invoice_customer_triples as
select 'customer' "rdf_property",
       ('invoice#' || id) "rdf_resource",
       customer "rdf_object"
       from test_invoices;
drop view MR_triples;
create view MR_triples as
   (select rdf_property, rdf_resource, rdf_object from
invoice_date_triples)
   union
   (select rdf_property, rdf_resource, rdf_object from
invoice_customer_triples)
   union
   (select rdf_property, rdf_resource, rdf_object from
MR_triples_base);
```

[1109] This will result in the following additional triples being available from the MR:

rdf_resource	rdf_property	rdf_object
invoice#1	last_updated	10-JAN-99
invoice#1	customer	Ford
invoice#2	last_updated	25-FEB-99
invoice#2	customer	Cisco

[1110] The disadvantage to this schema is that it is not normalized and stores a tremendous amount of duplicate

Lipkin, p. 71, col. 1, lines 27 – 67

data. Many values for rdf resource and rdf property will be duplicated, since the same resource will have a number of properties, and property names will come from a well-known set.

[1111] RQLParser

[1112] RQLParser parses an RQL document and builds an execution plan for the query. The plan consists of a tree of Java classes called "Operators," where each Operator is responsible for returning a Vector of matching resources.

[1113] The Operator interface is defined as follows:

```
public interface Operator
{
/**
 * An operator knows how to return a Vector of matching
resource values
 * (typically URIs).
 * @param conn JDBC connection to the MR
 * @param targetRDF Target RDF file.
 * @return Vector of matching resources
 * @exception SQLException Thrown on a database error
 */
  public Vector getResources(Connection conn, String
targetRDF) throws SQLException, ParseException;
} /* Operator */
```

Lipkin, p. 71, col. 2, lines 1 - 25

Specifically, referring to the paragraph bridging columns 1 – 2, above, Lipkin states, "[t]he disadvantage to this schema is that it is not normalized and stores a tremendous amount of duplicate data. Many values for rdf resource and rdf property will be duplicated, since the same resource

will have a number of properties, and property names will come from a well known set." This passage thus does not, contrary to the Examiner's assertion, teach or suggest the recitation of the clause against which it has been cited.

Nor, with respect to that part of claim 53 which recites "merging into a bag triplets determined to be redundant on a basis of that confidence level" do the passages from Lipkin cited by the Examiner provide relevant teachings. The cited passages, page 19, col. 2, lines 14 – 22, and page 66, col. 1, lines 10 – 21, are reprinted below. These do not teach removing redundancy. Nor do they teach merging redundant triplets into a bag (or container or any other "collection"):

> [0385] The bean is automatically saved to the persistent store when it is created by a client application using the create() method, and when the container decides to synchronize the bean's state with the database if the bean's data has been changed by the client application. The container's decision is based on such factors as transactions, concurrency, and resource management. The container will remove the data from persistent store when the remove() method is called by a client on an entity bean.

Lipkin, p. 19, col. 2, lines 14 – 22

> [1023] Property
>
> [1024] The Property element identifies a specific, named property of a Resource. Its contents identify the named property (also known as the predicate). Its contents can be a nested property, that is, multiple property names separated by forward slashes. This syntax may navigate over multiple properties, where each property value is a resource with its own properties. This may be the same syntax used by RDF Query's "path" attribute for nested queries.
>
> [1025] As a convenience, it may not be necessary to specify Container-related properties as part of the path, that is, Bag, Seq, Alt, and li elements are automatically navigated past.

Lipkin, p. 66, col. 1, lines 10 - 21

As evident upon review of these passages, there is simply nothing in them that pertains to redundancy reduction — much less, removing redundancy based, in part, on merging triplets determined to be redundant into a bag or other collection.

Likewise, with respect to that part of claim 53 which recites "comparing sequential levels of objects of the RDF triples," the passage from Lipkin cited by the Examiner provides no relevant teachings. The cited passage, page 9, col. 2, lines 43 – 67, is reprinted below. This does not teach removing redundancy based on comparing sequential levels of objects of RDF triples, or otherwise. Indeed, although the passage uses the word "object" and displays a table with levels, nothing it states has to do with removing redundancy, at levels or otherwise:

[0241] As an example, the following are the values for a class of business object representing domains:

id	ui_name	description	cnumber	insert_spid
ddcls000000 000001005	Domain	Hierarchial Domain	195	10560
update_spid	delete_spid	sel_dat_spid	finder_id	fixed_attr_ct
10562	10561	10563	15710	14
attr_ct	flags	next_attr_cnum	prefix	table_name
14	1100001100	100000	domin	fgt_domain
domain_cnum	java_class_name		hlevel	parent_id
	com.saba.busobj.SabaDomain		1	

[0242] 2. fgt_dd_attr

[0243] The attributes of each class of business object is stored in this table. This table also describes basic properties of each attribute.

Lipkin, p. 9, col. 2, lines 47 – 67

In view of the foregoing, the 35 USC 102 rejection of claims 53 lacks basis in fact and law. That rejection should be withdrawn.

Claims 27 – 30, 34, 39 – 41 and 43 – 44 are Patentable over Lipkin + Bradford + Delcambre

Independent claim 27 is directed to digital data processing methods for enterprise application integration which include storing, in a data store, RDF triplets representing transactional information received from each of a plurality of databases. The claim was previously amended to recite displaying on a browser a markup language document — which ***browser markup language document*** generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and which presents via the browser content generated from the data store in response to those queries.

The cited references fail to teach such a use of a markup language document.

The failings of Lipkin in this regard were pointed out by Applicants in their past response, as well as at Interview. However, the Examiner has attempted to re-read claim 27 onto that reference, urging that Applicant's recitation "markup language documents" reads on Lipkin's XML. *See,* Office Action at page 6, first paragraph. That misses the point.

The Applicants do not deny that Lipkin uses XML. They deny that Lipkin's XML equates with the markup language document recited in claim 27, specifically, a markup language document that:

(i) is displayed on a browser,

(ii) generates one or more queries for application to the data store,

(iii) in response to one or more user selections and/or responses to user-input controls specified ***by the document***, and

(iv) presents, via the browser, content generated from the data store in response to those queries.

Lipkin's XML does little if any of this! And, although the Examiner cites passage to the contrary, review of those passages reveal that they have little to do with this topic. For example, the cited passage at page 73, col. 2, lines 1 – 25 does <u>not</u> teach use of a markup language to generate queries. It merely discusses the use of XML to transfer messages from a so-called Business Server to so-called Interface Server for display in HTML or on Palm Pilots (or other PDAs). Nowhere does Lipkin mention *using the XML* to generate queries:

> [0024] In another embodiment, the present invention provides a network node in a network having a user node including a browser program or other general-purpose or purpose-built client program, coupled to the network, the user node providing information and requests for information, and providing application related commands on the network, the network node including a server node responsive to a request from the user node to process data for generating web content, whereby the server node provides a first mechanism for generating a plurality of tasks required to separate data production elements, interactive elements, and display elements, and a second mechanism for generating the web content based on the separated elements.

Put another way, unlike Applicant's recited markup language document, Lipkin's XML is merely the messenger — not the means. Applicant's recited markup language document, on the other hand is the *means*, i.e., the means for generating queries.

These shortcomings likewise differentiate Lipkin from pending independent claims 39 and 43, which also recite a browser markup language document generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-

input controls specified by the document and that presents via the browser content generated from the data store in response to those queries.

Neither Bradford nor Delcambre provide teachings that remedy these shortcomings of Lipkin. Thus, neither of those secondary references teach or suggest a browser markup language document generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and that presents via the browser content generated from the data store in response to those queries.

Indeed, the Examiner cites Bradford only for the proposition that transactional information can be stored in RDF. And, the Examiner cites Delcambre for the proposition that a data query can be stored. However, none of those teachings rises to the level of a browser markup language document of the type recited by Applicant in each of independent claims 27, 39 and 43.

For these reasons, among others, Lipkin, Bradford and Delcambre fail individually and together to suggest the subject matter of independent claims 27, 39 and 43, as well as that of the dependent claims which recite further limitations thereon.

Conclusion

This responds in full to the final Office Action in the above-cited case. The claims are amended for form and shown to be patentably distinct from the cited art. In view hereof, the Applicants request that the rejection be withdrawn and that application be passed forward to issuance.

Respectfully submitted,
NUTTER, McCLENNEN & FISH, LLP

David J. Powsner
Reg. No. 31,868

Attorney for Applicant
World Trade Center West
155 Seaport Boulevard
Boston, MA 02210-2604
Tel: (617)439-2717
Fax: (617)310-9717